LA-UR- 99-3959

Approved for public release; distribution is unlimited.

Title: User Documentation for defaultxs.cpp

Author(s): Travis K. Gray

Submitted to: Documentation for a utility code for use with MCNP.

LA-UR-99-3959

Los Alamos
NATIONAL LABORATORY
memorandum

Applied Theoretical & Computational Physics Division
Code Integration Group, XCI
Los Alamos, New Mexico 87545

To/MS:	Distribution
From/MS:	T. Gray, XCI / MS F663
Phone/FAX:	7-9145 / 5-5553
Symbol:	XCI:TKG-99-53(U)
Date:	August 4, 1999

SUBJECT: User Documentation for *defaultxs.cpp*

The utility code *defaultxs.cpp* is used to generate a listing of the default cross-sections used by MCNP[1] for a specified *xsdir* file. An MCNP user specifies the data file he wishes to use for a given nuclide by the ZAID entries on the material cards in the input deck. The user may use a full ZAID specification such as 13027.50c to indicate the use of the continuous-energy neutron data for ^{27}Al referenced by the ending of '.50'. MCNP also allows the use of a partial ZAID such as 13027 for ^{27}Al. In this instance the user will get the default cross-section for 13027 as specified by the user's specific *xsdir* file. To determine the default cross-section for ^{27}Al, MCNP will search the *xsdir* file to find the first occurrence of '13027' appropriate for the particular problem (such as neutron data for a mode n or mode np problem). As each user may have a different ZAID ordering in their *xsdir* file, they will obtain different cross-sections when using the default option. Use of default cross-sections is not recommended for this reason. Users should specify the appropriate cross-section choice for their particular application. More information is available in the MCNP manual on the *xsdir* file and default cross-sections in Appendix F and Chapter 2 of the MCNP manual respectively.

1 General Information

defaultxs.cpp is designed to search through a user specified *xsdir* file and produce a listing of default cross-sections. The code will interactively query the user whether to generate default cross-section listing for an entire *xsdir* file, one specific data type in that *xsdir* file, or a specific nuclide in that *xsdir* file. The code will also query the user whether to output the information to the screen or to a user-specified output file.

2 Compiling the Source Code

defaultxs.cpp is coded in C++. The compiler used during development was the GNU compiler available for most UNIX systems and free of charge (http://www.gnu.org). There has been no attempt to compile *defaultxs.cpp* under Windows or the Macintosh Operating Systems. The primary operating system used during development was

[1] J. R. Briesmeister, Editor, "MCNP – A General Monte Carlo N-Particle Transport Code," Los Alamos National Laboratory Report, LA-12625-M (March 1997).

Solaris. There are a few machine dependent aspects of the code associated with the screen output, but these will not affect the results.

Example compile commands:

> **g++ defaultxs.cpp or c++ defaultxs.cpp** to create *a.out* as the executable
> **g++ defaultxs.cpp –o defaultxs** to create *defaultxs* as the executable

3 Input

As mentioned previously, the code will interactively query the user for the necessary information to generate the default cross-section listing. There are three types of listings that *defaultxs.cpp* can generate: (1) a listing containing default cross-sections for all types of data contained in the *xsdir* file, (2) a listing for a specific type of data such as continuous-energy neutron data, or (3) a listing for a specific nuclide. The types of data MCNP can utilize are listed in Table 1. Based on the response, the code will query the user for the specific type of data or the ZAID for options (2) or (3). When specifying a ZAID, it must be in the form of ZZZAAA. For neutrons, deuterium with an atomic number of 1 and a mass number of 2 has a ZAID of 1002. However, for photons, the ZAID of Aluminum with an atomic number of 13 is 13000. For option (3) you are then asked only to specify this ZAID and the data type. The types of data that can be specified for both options (2) and (3) are given in Table 1.

Table 1: MCNP Data Types

Data Type	Abbreviated Notation
continuous energy neutron tables	c
discrete energy tables	d
dosimetry tables	y
S(a,b) thermal tables	t
continuous energy photon tables	p
continuous energy electron tables	e
multi-group neutron tables	m
multi-group photon tables	g

The code itself only searches on the abbreviated notation. If you select a data type denoted by ‘k’, which does not exist in the file you specified, then the code simply returns that no match was found. However, if the data exists in the *xsdir*, it does not matter if the type is not listed in the above table. For example, in the future MCNP will have the capability to use continuous-energy proton tables (h) or photonuclear tables. If the data is in the *xsdir* file, the code will generate a listing for the indicated data type. However, if the data type is not listed in the above table, or is not photonuclear or proton data, the output of that specific data type will not be sorted according to data type. This will only be obvious if there exists multiple alien data types in the *xsdir* file that the user searches.

Next, the code queries the user for the *xsdir* file to search. The user can specify the *xsdir* file in their **CURRENT WORKING DIRECTORY**, or the full path and filename for a *xsdir* file. Without the full path and filename, UNIX believes the file is in the root directory. Another caveat is to not use the tilde (~) convention from UNIX when specifying the path. This is not recognized by the code and will lead to an infinite loop with no file to read. An example is included in the sample session in section 6.

4 Search

The search routine will gather the appropriate ZAID's and data types from the ZZZAAA.lib at the beginning of each entry. It will compare new ZAID's and data types to those already noted as being the first occurrence of that ZAID and data type. If this particular combination has not occurred as yet in the program execution, then it will be assigned as a default data library and be outputted at the program execution end. It should also be noted, if you're using the *xsdir* file that is provided with MCNP, or any file of similar length, a total search of the file could take up to 10 minutes on a Sun Ultra 10.

5 Output

After the search if completed, the code queries the user for the type of output to generate. The user may specify to write the output to the screen or to a file. If the user selects the option to write to an output file, the code will then query the user for the filename. The strictness of giving a pathname is not necessary here, as the code will create the file in the program's current working directory. The same format will be used either when output is sent to the screen or to a file. However, the screen output will include a date and time of the program execution.

The sorting done by the program utilizes the Unix shell commands '*grep*' and '*sort*'. The output is grouped only for the data types listed in Table 1 as well as photonuclear and proton data. All other data types are considered foreign to the code and will only be sorted numerically by ZAID in the output.

6 Example Session

```
keanu|tgray|1> g++ defaultxs.cpp
keanu|tgray|2> ./a.out
```

(Screen Cleared)

```
Default Cross Section Values in the XSDIR of MCNP

This program is designed to search the user's xsdir file used by MCNP
and find out the cross section data library used for each isotope.
```

```
This can be done either by the user specifying an XSDIR file to search
or allowing the program to open the XSDIR file in the user's
CURRENT WORKING DIRECTORY

The user will also have the options of searching the entire file
for every isotope, search for a specific data type in the file, or
just a single isotope and have the option of outputting the data
either to a file or to the screen.

Press return to continue. . .
```

(Screen Cleared)

```
Do you wish to :
     1.  Search the entire XSDIR file
     2.  Search the xsdir file for a specific data type
     3.  Search only for a specific isotope / ZAID in the XSDIR file
What is your choice (1/2/3)? 1

What XSDIR file do you want to search???
     1. Enter the pathname to your file (path and filename)
     2. Select the XSDIR file in your CURRENT WORKING DIRECTORY

What option do you wish to choose (1/2)? 1

What is the entire path and filename for your XSDIR file? ./xsdat1

This could take several minutes. . .

Do wish you wish to output the data to the screen or a file ???
     1. file output
     2. screen output
What is your choice? 2

Fri Jul  9 11:50:34 MDT 1999
ZAID             Data Library
----------------------------------
1001.60c         endf601
1002.55c         rmccs1
2004.42c         endl921
6000.60c         endf601
6012.21c         100xs1
23000.51c        rmccs1
1001.50d         drmccs1
5011.56d         newxsd1
```

```
Do you wish to search for another Isotope or XSDIR file (yes/no)? no

End of Program Execution
keanu|tgray|3>
```

7 Use of an Input Deck

The user may wish to use an input deck to enter the options for the code. This will also facilitate multiple searches where the user will not be forced to type numerous options and ZAIDs. The format for an input deck would be as follows, if the deck is to be named xsinput and following the same input as in the example session (NOTE: text in bold would actually appear in the input deck while text in italics are explanations of the items in bold).

```
keanu|tgray|4>g++ defaultxs.cpp
keanu|tgray|5>less xsinput
```

A new line must be the first item in the deck to accommodate the “Press Return to Continue”.

1	*First Option (search entire xsdir file.)*
1	*Second Option (Specify a xsdir)*
./xsdat1	*Path and filename for specifying a xsdir file*
2	*Third Option (screen output)*
no	*No further files or ZAID’s to be processed*

```
keanu|tgray|6>./a.out < xsinput
```

If a user wanted to run the code for multiple ZAID’s, the input deck would look at follows:

A new line must be the first item in the deck to accommodate the “Press Return to Continue”.

1	*First Option (search entire xsdir file.)*
1	*Second Option (Specify a xsdir)*
./xsdat1	*Path and filename for specifying a xsdir file*
2	*Third Option (screen output)*
yes	*Further files to be processed*
2	*First Option(search xsdir for one data type)*
2	*Second Otion (Use the default xsdir file)*
c	*Data type to search default xsdir file for*
1	*Third Option (file output)*
cxsdir	*File to Output data to*
yes	*Further files to be processed*
3	*First Option (Specify one ZAID)*
1	*Second Option (Specify a xsdir)*
./xstest	*Path and filename for specifying a xsdir file*
1001	*Search xstest for H-1*
d	*Search for the first H-1 with the d data type*

`1`	*Third Option (file output)*
`h1d`	*File to Output data to*
`no`	*No further files to process*

It should be stated that while these examples used screen output; sending the output to the screen is not recommended when using an input deck. The output scrolls by very quickly and often gets run together if the user tries to scroll back up. Also, when the output is sent to a file, the file is created each time. So, if the user tries to send all the output from multiple processes to one file, each successive process will over write the previous output. The easiest solution is to use different files and concatenate them if necessary.